| Question | Part | Marking guidance | Total marks |
|---|---|---|---|
| 01 | 1 | **Mark is for AO1 (understanding)**<br><br>**C** Only two of the examples of code are in low-level languages;<br>**If more than one lozenge shaded then mark is not awarded** | 1 |
| 01 | 2 | **4 marks for AO1 (understanding)**<br><br>Maximum four marks from:<br><br>• High-level languages have built-in functions;<br>• High-level languages have built-in libraries;<br>• High-level languages have more support/help;<br>• High-level languages have structures (such as selection and iteration);<br>• High-level languages can be less machine dependent/more portable;<br>• It (usually) requires fewer lines of code to be written;<br>• It is (usually) quicker to develop code in high-level languages;<br>• It is easier to find mistakes in code;<br>• The code is easier to maintain//understand;<br>• It is easier to structure code in high-level languages;<br><br>**NE.** references to efficiency or speed unless correctly qualified;<br>**A.** Easier to read in place of easier to understand on this occasion;<br>**R.** Answers relating to programmer expertise; | 4 |

| Question | Part | Marking guidance | Total marks |
|---|---|---|---|
| 01 | 3 | **2 marks for AO1 (understanding)**<br><br>[Statement A:] compiler;<br>[Statement B:] assembler; | 2 |

| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 02 | 1 | **Mark is for AO2 (apply)**<br><br>**B**: Integer;<br><br>**R.** If more than one lozenge shaded. | 1 |
| 02 | 2 | **1 mark for AO2 (apply)**<br><br>Boolean/bool; | 1 |
| 02 | 3 | **3 marks for AO2 (apply)**<br><br>1 mark for each correct value of `valid`;;;<br><br>**Value of** `instr` / **Final value of** `valid`:<br>`ADD R0, R1` – `False`<br>`ADD: R0, R1` – `True`<br>`HALT` – `True` | 3 |
| 02 | 4 | **Mark is for AO1 (understanding)**<br><br>Machine code;<br><br>**A.** binary;<br>**A.** object code; | 1 |
| 02 | 5 | **2 marks for AO1 (understanding)**<br><br>**Max 2 marks from:**<br><br>(High-level languages) are better supported;<br>(High-level languages) provide built-in subroutines;<br>(High-level languages) provide programming structures such as iteration and selection;<br>(Code written in high-level languages) is normally shorter;<br>(High-level languages) allow creation of subroutines;<br>(High-level languages) provide data structures;<br>(High-level languages) are easier to understand/read;<br>(High-level languages) are easier to debug;<br><br>**A.** any other correct justification. | 2 |

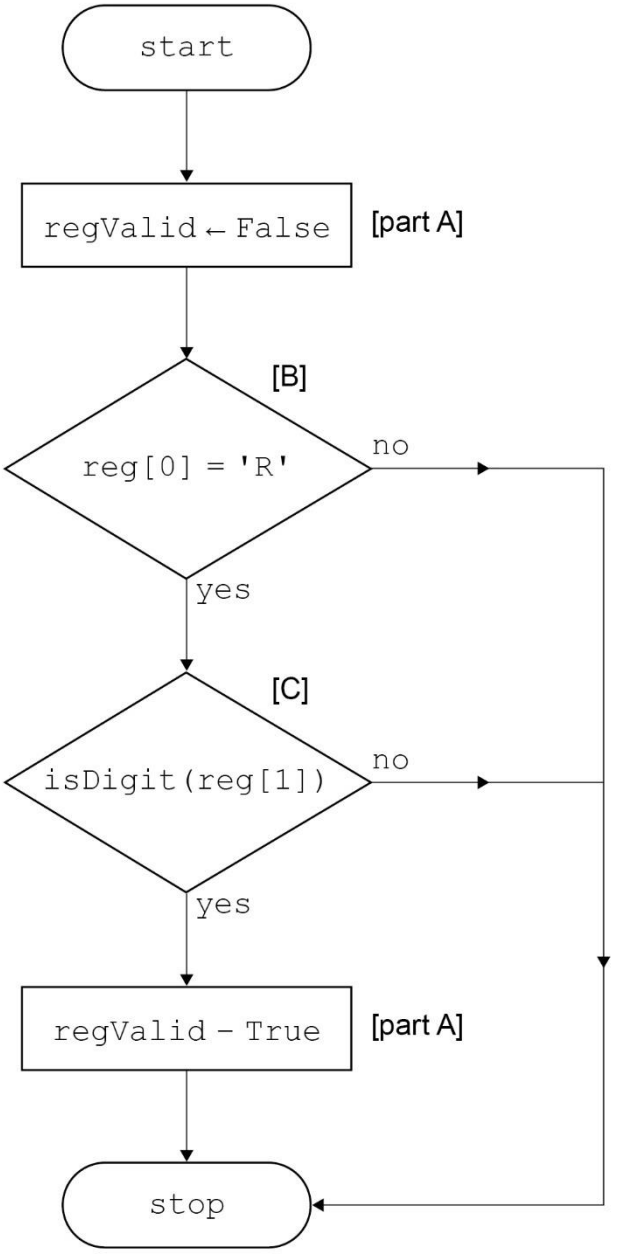
| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 02 | 6 | **3 marks for AO3 (program)**<br><br>**Mark A** for setting the variable `regValid` to `True`/`False` within a selection structure;<br>**Mark B** for using a Boolean condition that checks if the first character is an `'R'`;<br>**Mark C** for using a Boolean condition that checks if the second character is a digit;<br><br>**Max 2 marks** if any errors in the answer.<br><br>**A.** minor changes to variable identifiers if the meaning is still clear.<br><br>Example of fully correct answer:<br><br>`regValid ← False` [part A]<br>`IF reg[0] = 'R' and isDigit(reg[1]) THEN` [B,C]<br>`  regValid ← True` [part A]<br>`ENDIF`<br><br>Example of another fully correct answer:<br><br>`IF reg[0] = 'R' THEN` [B]<br>`  IF isDigit(reg[1]) THEN` [C]<br>`  regValid ← True` [part A]<br>`  ELSE`<br>`  regValid ← False` [part A]<br>`  ENDIF`<br>`ELSE`<br>`  regValid ← False` [part A]<br>`ENDIF`<br><br>Example of 2 mark answer:<br><br>`IF reg[0] = 'R' or isDigit(reg[1]) THEN` [B,C]<br>`  regValid ← True` [part A]<br>`ELSE`<br>`  regValid ← True` [part A]<br>`ENDIF`<br><br>(only 2 marks awarded as the answer contains an error in the Boolean condition) | 3 |

Example of another 2 mark answer:

```
IF reg[0] = 'R' and isDigit(reg[1]) THEN [B,C]
     regValid ← True                     [part A]
ENDIF
```

(only 2 marks awarded as only part of mark A is given)

Example of a fully correct flowchart solution:



| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 03 | | **2 marks for AO1 (understanding)**<br><br>**C** The code is not translated using a compiler;<br><br>**E** The code can directly manipulate computer registers;<br><br>**R.** if more than **two** lozenges shaded | 2 |

| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 04 | 1 | **Mark is for AO1 (recall)**<br><br>To convert / translate / change / turn assembly language into machine code;<br><br>**NE.** convert/translate assembly language | 1 |

| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 04 | 2 | **4 marks for AO1 (understanding)** | 4 |

| Level | Description | Mark Range |
|---|---|---|
| 2 | At the upper end of the mark range the student will have shown a coherent and accurate explanation of how an interpreter works with clear references to the principles of operation.<br><br>At the lower end of the mark range the student will have demonstrated a good understanding of how interpreters function. | 3–4 |
| 1 | At the upper end of the mark range the student will have included some explanation of how an interpreter works though there may be key steps or elements either missing or described inaccurately.<br><br>At the lower end of the mark range the student will have made one or two relevant statements but there may be errors or omissions in their understanding of how interpreters function. | 1–2 |
| | No creditworthy material. | 0 |

**Indicative content**

- Interpreters do not produce any machine code so the program needs to be translated each time it is executed.
- They call machine code subroutines within their own code to carry out commands / directly execute the instruction.

| | | | |
|---|---|---|---|
| | | • Translating a line / statement at a time (rather than all at once) which it then executes.<br>• If a runtime error is found the interpreter stops. | |

<table>
<tr><th>Qu</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>05</td><td>1</td><td>2 marks for AO1 (understanding)<br><br>
<table>
<tr><th></th><th>A, B or C</th></tr>
<tr><td>Assembly language</td><td>B</td></tr>
<tr><td>High-level language</td><td>A</td></tr>
<tr><td>Machine code</td><td>C</td></tr>
</table>
<br>Mark as follows:<br><br>1 mark for one correct row;<br>2 marks if all correct;;<br><br>R. repeated letters</td><td>2</td></tr>
</table>

| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 05 | 2 | **Mark is for AO1 (understanding)**<br><br>**Maximum 1 mark for any of the following:**<br><br>• Programs written in assembly language run faster // use less processor time;<br>• Programs written in assembly language can interact directly with hardware (when executing);<br>• Assembly language programs require less memory (when executing);<br>• Programs written in assembly language have no unnecessary code added by a compiler;<br><br>**A.** assembly language requires less translation<br>**A.** the programmer can take advantage of knowledge about the program / problem that isn't available to the compiler<br>**A.** assembly language is faster to translate | 1 |

| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 05 | 3 | **Mark is for AO1 (understanding)**<br><br>**A** A compiler translates all the original program code before execution.<br><br>**R.** if more than one lozenge shaded | 1 |

| Qu | Part | Marking guidance | Total marks |
|---|---|---|---|
| 06 | 1 | **Mark is for AO1 (understanding)**<br><br>**C** Only two of the examples of code are in low-level languages;<br><br>**R.** If more than one lozenge shaded | 1 |
| 06 | 2 | **4 marks for AO1 (understanding)**<br><br>Maximum four marks from:<br><br>• High-level languages have built-in functions;<br>• High-level languages have built-in libraries;<br>• High-level languages have more support/help;<br>• High-level languages have structures (such as selection and iteration);<br>• High-level languages can be less machine dependent/more portable;<br>• It (usually) requires fewer lines of code to be written;<br>• It is (usually) quicker to develop code in high-level languages;<br>• It is easier to find mistakes in code;<br>• The code is easier to maintain//understand;<br>• It is easier to structure code in high-level languages;<br><br>**NE.** references to efficiency or speed unless correctly qualified;<br>**A.** Easier to read in place of easier to understand on this occasion;<br>**R.** Answers relating to programmer expertise; | 4 |
| 06 | 3 | **2 marks for AO1 (understanding)**<br><br>[Statement A:] compiler;<br>[Statement B:] assembler; | 2 |

| Question | Part | Marking guidance | Total marks |
|---|---|---|---|
| **07** | | **2 marks for AO1 (understanding)**<br><br>**A** Machine code is directly executed by the processor;<br>**E** Machine code is specific to a family of processors;<br><br>**R.** If more than two lozenges shaded | 2 |

| Question | Part | Marking guidance | Total marks |
|---|---|---|---|
| 08 | | **3 marks for AO1 (understanding)**<br><br>**Maximum of three** marks from:<br><br>MP1. HLL has higher level of abstraction;<br>MP2. HLL runs across different CPU families / is portable;<br>MP3. HLL is English-like / easier to learn / understand / debug;<br>**R.** (HLL are) easier to read<br>MP4. HLL may require a compiler to translate the code / create executable program;<br>MP5. HLL may require an interpreter to run (the program);<br>MP6. HLL commands equate to multiple machine code commands;<br>MP7. HLL programs provide built-in subroutines;<br>MP8. Programs written in LLL are (usually) more efficient;<br>MP9. Programs written in LLL (usually) use less memory;<br>MP10. Programs written in LLL can directly access hardware components;<br><br>**R.** LLL don't need to be translated before execution | 3 |

| Question | Part | Marking guidance | Total marks |
|---|---|---|---|
| 09 | | **6 marks for AO1 (understanding)**<br><br>**Note for examiners:**<br>**Maximum of three** marks if only **one** of compiler or interpreter is described<br><br>**Maximum of six** marks from:<br><br>Compiler<br>MP1. Translates/compiles the whole program (in one go);<br>MP2. Standalone program does not require the compiler / source code on computer to run;<br>MP3. Produces a list of (syntax) errors after translation;<br>MP4. does not execute the programming code // creates an executable file / object code / machine code;<br>Interpreter<br>MP5. Translates/executes one line at a time;<br>MP6. Program requires interpreter to be present (on computer) to run the code;<br>MP7. Requires source code to be present/translated each time program is run // no separate executable file is created;<br>MP8. Stops execution/displays an error message as soon as a (syntax) error is encountered;<br>MP9. Executes the programming code as it is being translated; | 6 |